

[illegible]

EXE

Mod

EDT

ED

ED
EDED
EDED
EDED
ED

ED

ED

ED

ED

ED
EDSYN
LBA

110

[illegible]


```
0001 0 %TITLE 'EDT$SCRNEWINS - insert a line on the screen'
0002 0 MODULE EDT$SCRNEWINS (
0003 0 IDENT = 'V04-000'
0004 0 ) =
0005 1 BEGIN
0006 1
0007 1 *****
0008 1 *
0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0011 1 * ALL RIGHTS RESERVED.
0012 1 *
0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0018 1 * TRANSFERRED.
0019 1 *
0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0022 1 * CORPORATION.
0023 1 *
0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0026 1 *
0027 1 *
0028 1 *****
0029 1
0030 1
0031 1 ++
0032 1 FACILITY: EDT -- The DEC Standard Editor
0033 1
0034 1 ABSTRACT:
0035 1
0036 1 This module updates the screen information data structure to
0037 1 reflect the insertion of a line.
0038 1
0039 1 ENVIRONMENT: Runs at any access mode - AST reentrant
0040 1
0041 1 AUTHOR: Sharon M. Burlingame, CREATION DATE: September 15, 1982
0042 1
0043 1 MODIFIED BY:
0044 1
0045 1 1-001 - Original. SMB 15-Sep-1982.
0046 1 1-002 - Remove the position parameter and assume EOB is last line. SMB 06-Oct-1982
0047 1 1-003 - Remove setting of EDT$SA_SCR_PTR. SMB 10-Oct-1982
0048 1 1-004 - Add support for NOTRUNCATE mode. JBS 11-Oct-1982
0049 1 1-005 - Return the number of lines produced. JBS 11-Oct-1982
0050 1 1-006 - Fix a bug found by the QA system in finding a deleted line. JBS 12-Oct-1982
0051 1 1-007 - In NOTRUNCATE mode, leave room for the blob in lines after the first. JBS 12-Oct-1982
0052 1 1-008 - Fix call to EDT$FMT_CHWID. JBS 13-Oct-1982
0053 1 1-009 - Use memory allocation counter. SMB 18-Oct-1982
0054 1 1-010 - When deleting lines from the screen data base, delete complete records. JBS 23-Oct-1982
0055 1 1-011 - If we delete the [EOB] line from the screen data base, clear the EOB pointer. JBS 23-Oct-1982
0056 1 1-012 - Allow up to five times the number of lines on the screen of screen data blocks. JBS 24-Oct-1982
0057 1 1-013 - [EOB] can only be inserted at the end of the screen data base. JBS 24-Oct-1982
```

```
: 58      0058 1 | 1-014 - Correct the code that uses the earliest deleted line. JBS 25-Oct-1982
: 59      0059 1 | 1-015 - Fix another bug in the code that searches for the earliest deleted line. JBS 28-Oct-1982
: 60      0060 1 | 1-016 - Maintain SCR EDIT_MINPOS. JBS 28-Oct-1982
: 61      0061 1 | 1-017 - Use EDT$SCC_CNDEL to free a line block. JBS 29-Oct-1982
: 62      0062 1 | 1-018 - Add two parameters to help nottruncate inserts. SMB 03-Dec-1982
: 63      0063 1 | 1-019 - Change the handling of EDT$SG_SHF. JBS 14-Dec-1982
: 64      0064 1 | 1-020 - Fix tabbing at end of line in nottruncate mode. JBS 15-Dec-1982
: 65      0065 1 | 1-021 - Remove the edit buffer. JBS 27-Dec-1982
: 66      0066 1 | 1-022 - Set the rebuild flag if we delete the cursor line. JBS 03-Jan-1983
: 67      0067 1 | 1-023 - Always use a full size screen data base, so we won't lose any lines from the current record
: 68      0068 1 |
: 69      0069 1 | --
: 70      0070 1 |
```



```
.. 72 0071 1 %SBTTL 'Declarations'
.. 73 0072 1
.. 74 0073 1 | TABLE OF CONTENTS:
.. 75 0074 1 |
.. 76 0075 1
.. 77 0076 1 REQUIRE 'EDTSRC:TRAROUNAM';
.. 78 0515 1
.. 79 0516 1 FORWARD ROUTINE
.. 80 0517 1 | EDT$SC_LNINS;
.. 81 0518 1
.. 82 0519 1 |
.. 83 0520 1 | INCLUDE FILES:
.. 84 0521 1 |
.. 85 0522 1
.. 86 0523 1 REQUIRE 'EDTSRC:EDTREQ';
.. 87 0658 1
.. 88 0659 1 |
.. 89 0660 1 | MACROS:
.. 90 0661 1 |
.. 91 0662 1 | NONE
.. 92 0663 1 |
.. 93 0664 1 | EQUATED SYMBOLS:
.. 94 0665 1 |
.. 95 0666 1 |
.. 96 0667 1 LITERAL
.. 97 0668 1 | MAX_SCREEN_LINES = 22*5; ! Enough for 5 screens, so we can compute the number of lines above and below
.. 98 0669 1 |
.. 99 0670 1 |
100 0671 1 | OWN STORAGE:
101 0672 1 |
102 0673 1 | NONE
103 0674 1 |
104 0675 1 | EXTERNAL REFERENCES:
105 0676 1 |
106 0677 1 | In the routine
```

```
108 0678 1 %SBTTL 'EDT$SSC_LNINS - insert a line on the screen'
109 0679 1
110 0680 1 GLOBAL ROUTINE EDT$SSC_LNINS (
111 0681 1     | Insert a line on the screen
112 0682 1     | Location above which to insert
113 0683 1     | Address of new line
114 0684 1     | Length of new line
115 0685 1     |
116 0686 1     |
117 0687 1 ++
118 0688 1 FUNCTIONAL DESCRIPTION:
119 0689 1     A record has been inserted, update screen line information structure
120 0690 1     by indicating that the line(s) need to be repainted. Get memory
121 0691 1     from heap storage or the pool of available storage as needed.
122 0692 1
123 0693 1 FORMAL PARAMETERS:
124 0694 1
125 0695 1     SCRPTR             insert the new line(s) just before this line
126 0696 1
127 0697 1     REC_ADDR           the address of the new record
128 0698 1
129 0699 1     REC_LEN            the length of the new record
130 0700 1
131 0701 1 IMPLICIT INPUTS:
132 0702 1
133 0703 1     EDT$SG_MEM_CNT
134 0704 1     EDT$SA_WK_CN
135 0705 1     EDT$SZ_EOB_LN
136 0706 1     EDT$SA_CUR_BUF
137 0707 1     EDT$SA_FST_SCRPTR
138 0708 1     EDT$SA_LST_SCRPTR
139 0709 1     EDT$SA_CUR_SCRPTR
140 0710 1     EDT$SG_TI_WID
141 0711 1     EDT$SG_SHF
142 0712 1     EDT$SG_TRUN
143 0713 1
144 0714 1 IMPLICIT OUTPUTS:
145 0715 1
146 0716 1     EDT$SG_MEM_CNT
147 0717 1     EDT$SA_EOB_SCRPTR
148 0718 1     EDT$SA_FST_SCRPTR
149 0719 1     EDT$SA_CSR_SCRPTR
150 0720 1     EDT$SL_CUR_SCRLN
151 0721 1     EDT$SL_LNO_EMPTY
152 0722 1     EDT$SG_SCR_REBUILD
153 0723 1     EDT$SA_LST_SCRPTR
154 0724 1
155 0725 1 ROUTINE VALUE:
156 0726 1
157 0727 1     NONE
158 0728 1
159 0729 1 SIDE EFFECTS:
160 0730 1
161 0731 1     NONE
162 0732 1
163 0733 1 --
164 0734 1
```



```
165 0735 2 BEGIN
166 0736 2
167 0737 2 EXTERNAL ROUTINE
168 0738 2     EDT$FMT_MSG : NOVALUE,
169 0739 2     EDT$ALO_HEAP,
170 0740 2     EDT$FMT_CHWID,
171 0741 2     EDT$$$SC_LNDEL : NOVALUE;
172 0742 2
173 0743 2 EXTERNAL
174 0744 2     EDT$SG_MEM_CNT,
175 0745 2     EDT$SA_FST_AVLN : REF SCREEN_LINE,
176 0746 2     EDT$SA_WK_CN : REF LN_BLOCK,
177 0747 2     EDT$SZ_EOB_LN,
178 0748 2     EDT$SA_CUR_BUF : REF TBCB_BLOCK,
179 0749 2     EDT$SA_EOB_SCRPTR : REF SCREEN_LINE,
180 0750 2     EDT$SA_FST_SCRPTR : REF SCREEN_LINE,
181 0751 2     EDT$SA_LST_SCRPTR : REF SCREEN_LINE,
182 0752 2     EDT$SA_CUR_SCRPTR : REF SCREEN_LINE,
183 0753 2     EDT$SG_TRUN,
184 0754 2     EDT$SG_SHF,
185 0755 2     EDT$SG_TI_WID,
186 0756 2     EDT$SA_CSR_SCRPTR : REF SCREEN_LINE,
187 0757 2     EDT$SL_CUR_SCRLN : LN_BLOCK,
188 0758 2     EDT$SL_LNO_EMPTY : LN_BLOCK,
189 0759 2     EDT$SG_SCR_REBUILD;
190 0760 2
191 0761 2 MESSAGES ((INSMEM));
192 0762 2
193 0763 2 MAP
194 0764 2     SCRPTR : REF SCREEN_LINE;
195 0765 2
196 0766 2 LOCAL
197 0767 2     EDIT_CODE,
198 0768 2     NEW_SCRPTR : REF SCREEN_LINE,
199 0769 2     PREV_SCRPTR : REF SCREEN_LINE,
200 0770 2     INDX,
201 0771 2     NEED_ANOTHER_LINE,
202 0772 2     LEFT,
203 0773 2     RIGHT,
204 0774 2     CHAR,
205 0775 2     TXT,
206 0776 2     COL,
207 0777 2     LEN,
208 0778 2     NEW_ADDR;
209 0779 2
210 0780 2 !+
211 0781 2 !- Make sure the counter agrees with the data base.
212 0782 2
213 0783 2
214 0784 2 IF 0
215 0785 2 THEN
216 0786 2
217 0787 2     IF (.EDT$SA_FST_SCRPTR EQLA 0)
218 0788 2     THEN
219 0789 2         ASSERT (.EDT$SG_MEM_CNT EQL 0)
220 0790 2     ELSE
221 0791 2         BEGIN
```

Print a message
Allocate heap storage
Compute the width of a character
Free a line from the screen data base

Memory allocation count
List of free screen lines
Address of current workfile line
Address of EOB
Current text buffer control block
Pointer to EOB screen line
Pointer to first screen line info
Pointer to last screen line info
Pointer to the current screen line
1 = truncate mode
Screen shift amount
Width of the terminal
Current cursor line screen info
Absolute record number of that line
Special value for "empty" line number
1 = rebuild the screen data base from the work file

Repaint or insert
Address of new line info
Address of previous line info
Index for inserting multiple lines
1 = another line needed for this record
Left character position for this line
Right character position for this line

```
222 0792 3
223 0793
224 0794 LOCAL
225 0795 COUNT,
226 0796 SCRPTR1 : REF SCREEN_LINE,
227 0797 SCRPTR2 : REF SCREEN_LINE;
228 0798
229 0799 COUNT = 0;
230 0800 SCRPTR1 = .EDT$$A_FST_SCPTR;
231 0801 ASSERT (.SCRPTR1 [SCR_PRV_LINE] EQL 0);
232 0802 WHILE (.SCRPTR1 NEQA 0) DO
233 0803 BEGIN
234 0804 COUNT = .COUNT + 1;
235 0805 SCRPTR2 = .SCRPTR1;
236 0806 SCRPTR1 = .SCRPTR1 [SCR_NXT_LINE];
237 0807
238 0808 IF (.SCRPTR1 NEQA 0)
239 0809 THEN
240 0810 BEGIN
241 0811 ASSERT (.SCRPTR1 [SCR_PRV_LINE] EQLA .SCRPTR2);
242 0812 ASSERT (.SCRPTR1 NEQA .EDT$$A_FST_SCPTR);
243 0813 END;
244 0814
245 0815 END;
246 0816
247 0817 ASSERT (.SCRPTR2 EQLA .EDT$$A_LST_SCPTR);
248 0818 ASSERT (.COUNT EQL .EDT$$G_MEM_CNT);
249 0819 END;
250 0820
251 0821 INDX = 0; ! Start with first line
252 0822 NEED_ANOTHER_LINE = 1; ! Always need at least one line
253 0823 LEFT = 0; ! Start the first line at position 0
254 0824
255 0825 WHILE (.NEED_ANOTHER_LINE AND (.INDX LEQ 255)) DO
256 0826 BEGIN
257 0827
258 0828 !+ If we are inserting before a line, look for deleted lines; use the earliest
259 0829 !- one found.
260 0830
261 0831 IF (.SCRPTR NEQA 0)
262 0832 THEN
263 0833 BEGIN
264 0834 NEW_SCPTR = .SCRPTR [SCR_PRV_LINE];
265 0835
266 0836 IF (.NEW_SCPTR NEQA 0)
267 0837 THEN
268 0838 BEGIN
269 0839
270 0840 IF ((.NEW_SCPTR [SCR_EDIT_FLAGS] AND SCR_EDIT_DELLN) NEQ 0)
271 0841 THEN
272 0842 BEGIN
273 0843 LOCAL
274 0844 FOUND_EARLIEST;
275 0845
276 0846 FOUND_EARLIEST = 0;
277 0847
278 0848
```



```
279 0849 6
280 0850 6      WHILE ( NOT .FOUND_EARLIEST) DO
281 0851 7      BEGIN
282 0852 7      PREV_SCRPTR = .NEW_SCRPTR [SCR_PRV_LINE];
283 0853 7
284 0854 8      IF (.PREV_SCRPTR EQLA 0)
285 0855 7      THEN
286 0856 7      FOUND_EARLIEST = 1
287 0857 7      ELSE
288 0858 8      BEGIN
289 0859 8
290 0860 9      IF ((.PREV_SCRPTR [SCR_EDIT_FLAGS] AND SCR_EDIT_DELLN) NEQ 0)
291 0861 8      THEN
292 0862 8      NEW_SCRPTR = .PREV_SCRPTR
293 0863 8      ELSE
294 0864 8      FOUND_EARLIEST = 1;
295 0865 8
296 0866 7      END;
297 0867 7
298 0868 6      END;
299 0869 6
300 0870 6      EDIT_CODE = SCR_EDIT_MODIFY;
301 0871 6      END
302 0872 5      ELSE
303 0873 5      NEW_SCRPTR = 0;
304 0874 5
305 0875 4      END;
306 0876 4
307 0877 4      END
308 0878 3      ELSE
309 0879 3      NEW_SCRPTR = 0;
310 0880 3
311 0881 4      IF (.NEW_SCRPTR EQLA 0)
312 0882 3      THEN
313 0883 4      BEGIN
314 0884 4      EDT$G_MEM_CNT = .EDT$G_MEM_CNT + 1;      ! Incremented each time entered
315 0885 4      +
316 0886 4      Unless there is available memory, get some from heap storage for the
317 0887 4      new screen line information buffer.
318 0888 4      -
319 0889 4
320 0890 5      IF (.EDT$A_FST_AVLN EQLA 0)
321 0891 4      THEN
322 0892 5      BEGIN
323 0893 5
324 0894 6      IF ( NOT EDT$ALO_HEAP (%REF (SCR_SIZE), NEW_ADDR))
325 0895 5      THEN
326 0896 6      BEGIN
327 0897 6      EDT$G_SCR_REBUILD = 1;
328 0898 6      EDT$FMT_MSG (EDT$_INSMEM);
329 0899 6      RETURN (0);
330 0900 5      END;
331 0901 5
332 0902 5      NEW_SCRPTR = .NEW_ADDR;
333 0903 5      END
334 0904 4      ELSE
335 0905 5      BEGIN
```



```
336 0906 5 NEW_SCRPTR = .EDT$$A_FST_AVLN;  
337 0907 5 EDT$$A_FST_AVLN = .NEW_SCRPTR [SCR_NXT_LINE];  
338 0908 4 END;  
339 0909 4  
340 0910 4 EDIT_CODE = SCR_EDIT_INSLN;  
341 0911 4  
342 0912 4 !+ If the screen information structure is empty, then initialize  
343 0913 4 the screen pointers. Set up the first pointer as EOB if the  
344 0914 4 current line is at the end of the buffer.  
345 0915 4 !-  
346 0916 4  
347 0917 5 IF (.EDT$$A_FST_SCRPTR EQL 0)  
348 0918 4 THEN  
349 0919 5 BEGIN  
350 0920 5 EDT$$A_FST_SCRPTR = .NEW_SCRPTR;  
351 0921 5 EDT$$A_LST_SCRPTR = .NEW_SCRPTR;  
352 0922 5 NEW_SCRPTR [SCR_NXT_LINE] = 0;  
353 0923 5 NEW_SCRPTR [SCR_PRV_LINE] = 0;  
354 0924 5 END  
355 0925 4 ELSE  
356 0926 5 BEGIN  
357 0927 5 !+  
358 0928 5 Perform the insert above the given SCRPTR. If that SCRPTR is  
359 0929 5 zero, then insert the new line below the last screen pointer.  
360 0930 5 !-  
361 0931 5  
362 0932 6 IF (.SCRPTR EQLA 0)  
363 0933 5 THEN  
364 0934 6 BEGIN  
365 0935 6 NEW_SCRPTR [SCR_NXT_LINE] = 0;  
366 0936 6 NEW_SCRPTR [SCR_PRV_LINE] = .EDT$$A_LST_SCRPTR;  
367 0937 6 EDT$$A_LST_SCRPTR [SCR_NXT_LINE] = .NEW_SCRPTR;  
368 0938 6 EDT$$A_LST_SCRPTR = .NEW_SCRPTR;  
369 0939 6 !+  
370 0940 6 If we have exceeded our maximum screen line insert count, then  
371 0941 6 remove the top screen record.  
372 0942 6 !-  
373 0943 6  
374 0944 7 WHILE ((.EDT$$G_MEM_CNT GEQ MAX_SCREEN_LINES) AND !  
375 0945 6 (.EDT$$A_FST_SCRPTR NEQA .NEW_SCRPTR)) DO  
376 0946 7 BEGIN  
377 0947 7  
378 0948 7 DO  
379 0949 8 BEGIN  
380 0950 8 !+  
381 0951 8 Check for deleting the cursor line from the data base. If this happens we will  
382 0952 8 have to rebuild the screen data base.  
383 0953 8 !-  
384 0954 8  
385 0955 9 IF (.EDT$$A_CSR_SCRPTR EQLA .EDT$$A_FST_SCRPTR)  
386 0956 8 THEN  
387 0957 9 BEGIN  
388 0958 9 EDT$$A_CSR_SCRPTR = 0;  
389 0959 9 MOVELINE (EDT$$L_LNO_EMPTY, EDT$$L_CUR_SCRLN);  
390 0960 9 EDT$$G_SCR_REBUICD = 1;  
391 0961 8 END;  
392 0962 8
```



```
393 0963 8          EDT$SC_LNDEL (.EDT$A_FST_SCRPTR);
394 0964 8          END
395 0965 8          UNTIL ((.EDT$A_FST_SCRPTR [SCR_LINE_IDX] EQL 0) OR      !
396 0966 8          (.EDT$A_FST_SCRPTR EQLA .NEW_SCRPTR))
397 0967 8
398 0968 6          END;
399 0969 6
400 0970 6          END
401 0971 5      ELSE
402 0972 6          BEGIN
403 0973 6          PREV_SCRPTR = .SCRPTR [SCR_PRV_LINE];
404 0974 6          NEW_SCRPTR [SCR_PRV_LINE] = .PREV_SCRPTR;
405 0975 6          NEW_SCRPTR [SCR_NXT_LINE] = .SCRPTR;
406 0976 6
407 0977 7          IF (.PREV_SCRPTR NEQA 0)
408 0978 6          THEN
409 0979 6          PREV_SCRPTR [SCR_NXT_LINE] = .NEW_SCRPTR
410 0980 6          ELSE
411 0981 7          BEGIN
412 0982 7          EDT$A_FST_SCRPTR = .NEW_SCRPTR;
413 0983 7      !+
414 0984 7      ! If we have exceeded our maximum screen line insert count, then
415 0985 7      ! remove the bottom screen record.
416 0986 7      !-
417 0987 7
418 0988 8          WHILE ((.EDT$G_MEM_CNT GEQ MAX_SCREEN_LINES) AND      !
419 0989 7          (.EDT$A_LST_SCRPTR NEQA .NEW_SCRPTR)) DO
420 0990 8          BEGIN
421 0991 8
422 0992 8          DO
423 0993 9          BEGIN
424 0994 9      !+
425 0995 9      ! Check for deleting the cursor line from the data base. If this happens we will
426 0996 9      ! have to rebuild the screen data base.
427 0997 9      !-
428 0998 9
429 0999 10         IF (.EDT$A_CSR_SCRPTR EQLA .EDT$A_LST_SCRPTR)
430 1000 9         THEN
431 1001 10         BEGIN
432 1002 10         EDT$A_CSR_SCRPTR = 0;
433 1003 10         MOVELINE (EDT$L_LNO_EMPTY, EDT$L_CUR_SCRLN);
434 1004 10         EDT$G_SCR_REBUILD = 1;
435 1005 9         END;
436 1006 9
437 1007 9         EDT$SC_LNDEL (.EDT$A_LST_SCRPTR);
438 1008 9         END
439 1009 9         UNTIL ((.EDT$A_FST_AVLN [SCR_LINE_IDX] EQL 0) OR      !
440 1010 8         (.EDT$A_LST_SCRPTR EQLA .NEW_SCRPTR));
441 1011 8
442 1012 7         END;
443 1013 7
444 1014 6         END;
445 1015 6
446 1016 6         SCRPTR [SCR_PRV_LINE] = .NEW_SCRPTR;
447 1017 5         END;
448 1018 5
449 1019 4         END;
```



```
450 1020 4
451 1021 4      END;
452 1022 4
453 1023 4
454 1024 4      + Initialize the fields of the screen information block for repaint.
455 1025 4      -
456 1026 4      NEW_SCRPTR [SCR_EDIT_FLAGS] = .EDIT_CODE;
457 1027 4      NEW_SCRPTR [SCR_LINE_IDX] = .INDX;
458 1028 4      NEW_SCRPTR [SCR_EDIT_MINPOS] = 0;
459 1029 4      NEW_SCRPTR [SCR_EDIT_MAXPOS] = 255;
460 1030 4      NEW_SCRPTR [SCR_CHR_FROM] = .LEFT;
461 1031 4
462 1032 4      + Compute the right character position of the current line.
463 1033 4      -
464 1034 4
465 1035 4      IF ( NOT .EDT$G_TRUN)
466 1036 4      THEN
467 1037 4          BEGIN
468 1038 4              LOCAL
469 1039 4                  WIDTH;
470 1040 4
471 1041 4              IF (.INDX EQL 0)
472 1042 4              THEN
473 1043 4                  BEGIN
474 1044 4                      WIDTH = .EDT$G_TI_WID + .EDT$G_SHF;
475 1045 4                      COL = 0;
476 1046 4                      END
477 1047 4                  ELSE
478 1048 4                      BEGIN
479 1049 4                          WIDTH = .EDT$G_TI_WID + .EDT$G_SHF;
480 1050 4                          COL = .EDT$G_SHF + 2;
481 1051 4                          END;
482 1052 4
483 1053 4
484 1054 4      +
485 1055 4      - We must compute the width of each character to see how many will fit.
486 1056 4
487 1057 4      LEN = .REC_LEN - .LEFT;
488 1058 4      TXT = CH$PCUS (.REC_ADDR, .LEFT);
489 1059 4      RIGHT = .LEFT - 1;
490 1060 4      CHAR = CH$RCHAR_A (TXT);
491 1061 4
492 1062 4      WHILE ((.LEN GTR 0) AND ((.COL + EDT$FMT_CHWID (.CHAR, .COL)) LEQ .WIDTH)) DO
493 1063 4          BEGIN
494 1064 4              LEN = .LEN - 1;
495 1065 4              RIGHT = .RIGHT + 1;
496 1066 4              COL = .COL + EDT$FMT_CHWID (.CHAR, .COL);
497 1067 4              CHAR = CH$RCHAR_A (TXT);
498 1068 4              END;
499 1069 4
500 1070 4      IF (.LEN GTR 0)
501 1071 4      THEN
502 1072 4          BEGIN
503 1073 4
504 1074 4      + We need another line.
505 1075 4      -
506 1076 4          NEED_ANOTHER_LINE = 1;
```

```
507      LEFT = .RIGHT + 1;
508      END
509      ELSE
510      BEGIN
511      + We don't need another line.
512      -
513      NEED_ANOTHER_LINE = 0;
514      RIGHT = 255;
515      END;
516
517      END
518      ELSE
519      BEGIN
520      + Truncate mode, we never need more than one line.
521      -
522      NEED_ANOTHER_LINE = 0;
523      RIGHT = 255;
524      END;
525
526      NEW_SCRPTR [SCR_CHR_TO] = .RIGHT;
527
528      + If there is no current screen pointer, make this line the current line.
529      -
530      IF (.EDT$$A_CUR_SCRPTR EQLA 0) THEN EDT$$A_CUR_SCRPTR = .NEW_SCRPTR;
531
532      INDX = .INDX + 1;
533      END;
534
535      IF ((.EDT$$A_WK_LN EQLA EDT$$Z_EOB_LN) AND (.SCRPTR EQLA 0)) THEN EDT$$A_EOB_SCRPTR = .NEW_SCRPTR;
536
537      + Make sure the counter agrees with the data base.
538      -
539      IF 0
540      THEN
541      BEGIN
542      LOCAL
543      COUNT,
544      SCRIPT1 : REF SCREEN_LINE,
545      SCRIPT2 : REF SCREEN_LINE;
546
547      COUNT = 0;
548      SCRIPT1 = .EDT$$A_FST_SCRPTR;
549      ASSERT (.SCRIPT1 [SCR_PRV_LINE] EQL 0);
550
551      WHILE (.SCRIPT1 NEQA 0) DO
552      BEGIN
553      COUNT = .COUNT + 1;
554      SCRIPT2 = .SCRIPT1;
555      SCRIPT1 = .SCRIPT1 [SCR_NXT_LINE];
556
557      IF (.SCRIPT1 NEQA 0)
```



```
564      1134 4      THEN
565      1135 5      BEGIN
566      1136 5      ASSERT (.SCRPTR1 [SCR_PRV_LINE] EQLA .SCRPTR2);
567      1137 5      ASSERT (.SCRPTR1 NEQA .EDT$$A_FST_SCRPTR);
568      1138 4      END;
569      1139 4
570      1140 4      END;
571      1141 4
572      1142 4      ASSERT (.SCRPTR2 EQLA .EDT$$A_LST_SCRPTR);
573      1143 4      ASSERT (.COUNT EQL .EDT$$G_MEM_CNT);
574      1144 4      END;
575      1145 4
576      1146 4      !+ Return the number of lines produced.
577      1147 4      !-
578      1148 4      RETURN (.INDX);
579      1149 4      END;
580      1150 1
```

! of routine EDT\$\$\$SC_LNINS

.TITLE EDT\$SCRNEWINS EDT\$SCRNEWINS - insert a line on
the screen

.IDENT \V04-000\

```
.EXTRN EDT$$FMT_MSG, EDT$$ALO_HEAP
.EXTRN EDT$$FMT_CHWID, EDT$$$SC_LNDEL
.EXTRN EDT$$G_MEM_CNT, EDT$$A_FST_AVLN
.EXTRN EDT$$A_WK_CN, EDT$$Z_EOB_LN
.EXTRN EDT$$A_CUR_BUF, EDT$$A_EOB_SCRPTR
.EXTRN EDT$$A_FST_SCRPTR
.EXTRN EDT$$A_LST_SCRPTR
.EXTRN EDT$$A_CUR_SCRPTR
.EXTRN EDT$$G_TRUN, EDT$$G_SHF
.EXTRN EDT$$G_TI_WID, EDT$$A_CSR_SCRPTR
.EXTRN EDT$$L_CUR_SCRLN
.EXTRN EDT$$L_LNO_EMPTY
.EXTRN EDT$$G_SCR_REBUILD
.EXTRN EDT$_INSMEM, EDT$$INTER_ERR
```

.PSECT _EDT\$CODE, NOWRT, SHR, PIC, 2

```
.ENTRY EDT$$$SC_LNINS, Save R2,R3,R4,R5,R6,R7,R8,- : 0680
R9,R10,R11
SUBL2 #36, SP
MOVQ #1, NEED_ANOTHER_LINE : 0822
CLRL LEFT : 0823
MOVL SCRPTR, R11 : 0832
BLBS NEED_ANOTHER_LINE, 3$ : 0825
BRW 31$
CMLP INDX, #255
BGTR 2$
TSTL R11 : 0832
BEQL 7$
MOVL (R11), NEW_SCRPTR : 0835
BEQL 8$ : 0837
BBC #2, 13(NEW_SCRPTR), 7$ : 0841
CLRL FOUND_EARLIEST : 0848
BLBS FOUND_EARLIEST, 6$ : 0850
```

```
OFFC 00000
14 5E 24 C2 00002
AE 01 7D 00005
5B 04 AC D0 0000B
03 14 AE E8 0000F 1$:
000000FF 8F 18 0249 31 00013 2$:
AE D1 00016 3$:
F3 14 0001E
5B D5 00020
29 13 00022
58 6B D0 00024
26 13 00027
1F OD A8 02 E1 00029
14 50 D4 0002E
50 E8 00030 4$:
```

05	0D	5A	68	D0	00033	MOVL	(NEW_SCRPTR), PREV_SCRPTR	0852
			0A	13	00036	BEQL	5\$	0854
		AA	02	E1	00038	BBC	#2, 13(PREV_SCRPTR), 5\$	0860
		58	5A	D0	0003D	MOVL	PREV_SCRPTR, NEW_SCRPTR	0862
			EE	11	00040	BRB	4\$	
		50	01	D0	00042	5\$: MOVL	#1, FOUND_EARLIEST	0864
			E9	11	00045	BRB	4\$	0850
	04	AE	01	D0	00047	6\$: MOVL	#1, EDIT_CODE	0870
			02	11	0004B	BRB	8\$	0841
			58	D4	0004D	7\$: CLRL	NEW_SCRPTR	0879
			58	D5	0004F	8\$: TSTL	NEW_SCRPTR	0881
			67	12	00051	BNEQ	12\$	
		00000000G	00	D6	00053	INCL	EDT\$G_MEM_CNT	0884
		50 00000000G	00	D0	00059	MOVL	EDT\$A_FST_AVLN, R0	0890
			31	12	00060	BNEQ	10\$	
	20	AE	20	9F	00062	PUSHAB	NEW_ADDR	0894
			0E	D0	00065	MOVL	#14, 32(SP)	
			20	9F	00069	PUSHAB	32(SP)	
	00000000G	00	02	FB	0006C	CALLS	#2, EDT\$ALO_HEAP	
		17	50	E8	00073	BLBS	R0, 9\$	
	00000000G	00	01	D0	00076	MOVL	#1, EDT\$G_SCR_REBUILD	0897
		00000000G	8F	DD	0007D	PUSHL	#EDT\$ INSMEM	0898
	00000000G	00	01	FB	00083	CALLS	#1, EDT\$FMT_MSG	
			01F2	31	0008A	BRW	33\$	0899
		58	20	AE	D0	0008D	9\$: MOVL	NEW_ADDR, NEW_SCRPTR
			0B	11	00091	BRB	11\$	0890
		58	50	D0	00093	10\$: MOVL	R0, NEW_SCRPTR	0906
	00000000G	00	04	A8	D0	00096	MOVL	4(NEW_SCRPTR), EDT\$A_FST_AVLN
		04	AE	02	D0	0009E	11\$: MOVL	#2, EDIT_CODE
		00000000G	00	D5	000A2	TSTL	EDT\$A_FST_SCRPTR	0910
			13	12	000A8	BNEQ	13\$	0917
	00000000G	00	58	D0	000AA	MOVL	NEW_SCRPTR, EDT\$A_FST_SCRPTR	0920
	00000000G	00	58	D0	000B1	MOVL	NEW_SCRPTR, EDT\$A_LST_SCRPTR	0921
			68	7C	000B8	CLRQ	(NEW_SCRPTR)	0923
			00F0	31	000BA	12\$: BRW	23\$	0917
			58	D5	000BD	13\$: TSTL	R11	0932
			73	12	000BF	BNEQ	17\$	
		04	A8	D4	000C1	CLRL	4(NEW_SCRPTR)	0935
	50 00000000G	00	D0	D0	000C4	MOVL	EDT\$A_LST_SCRPTR, R0	0936
		68	50	D0	000CB	MOVL	R0, (NEW_SCRPTR)	
	04	A0	58	D0	000CE	MOVL	NEW_SCRPTR, 4(R0)	0937
	00000000G	00	58	D0	000D2	MOVL	NEW_SCRPTR, EDT\$A_LST_SCRPTR	0938
	0000006E	8F	00000000G	00	D1	000D9	14\$: CML	EDT\$G_MEM_CNT, #1T0
				D4	19	000E4	BLSS	12\$
		58	00000000G	00	D1	000E6	CML	EDT\$A_FST_SCRPTR, NEW_SCRPTR
				CB	13	000ED	BEQL	12\$
		59	00000000G	00	D0	000EF	MOVL	EDT\$A_FST_SCRPTR, R9
		59	00000000G	00	D1	000F6	15\$: CML	EDT\$A_CSR_SCRPTR, R9
				19	12	000FD	BNEQ	16\$
		00000000G	00	D4	000FF	CLRL	EDT\$A_CSR_SCRPTR	0958
00000000G	00	00000000G	00	06	28	00105	MOVC3	#6, EDT\$SL_LNO_EMPTY, EDT\$SL_CUR_SCRLN
		00000000G	00	01	D0	00111	MOVL	#1, EDT\$G_SCR_REBUILD
				59	DD	00118	16\$: R9	0960
		00000000G	00	01	FB	0011A	PUSHL	R9
				00	D0	00121	CALLS	#1, EDT\$SC_LNDEL
		59	00000000G	00	D0	00121	MOVL	EDT\$A_FST_SCRPTR, R9
			08	A9	95	00128	TSTB	8(R9)
			AC	13	0012B	BEQL	14\$	0965

58	59	D1	0012D	CMPL	R9, NEW_SCRPTR	0966					
	C4	12	00130	BNEQ	15\$	0946					
	A5	11	00132	BRB	14\$	0973					
5A	6B	D0	00134	17\$:	MOVL	(R11), PREV_SCRPTR	0974				
68	5A	7D	00137	MOVQ	PREV_SCRPTR, (NEW_SCRPTR)	0977					
	5A	D5	0013A	TSTL	PREV_SCRPTR	0979					
	06	13	0013C	BEQL	18\$	0982					
04	AA	58	D0	0013E	MOVL	NEW_SCRPTR, 4(PREV_SCRPTR)	0988				
		66	11	00142	BRB	22\$	0989				
00000000G	00	58	D0	00144	18\$:	MOVL	NEW_SCRPTR, EDT\$A_FST_SCRPTR	0999			
0000006E	8F	00000000G	00	D1	0014B	19\$:	CMPL	EDT\$G_MEM_CNT, #1T0	1002		
			52	19	00156	22\$	BLSS	EDT\$A_LST_SCRPTR, NEW_SCRPTR	1003		
	58	00000000G	00	D1	00158	22\$	CMPL	EDT\$A_LST_SCRPTR, R9	1004		
			49	13	0015F	20\$:	BEQL	EDT\$A_CSR_SCRPTR, R9	1007		
	59	00000000G	00	D0	00161	21\$:	MOVL	EDT\$A_CSR_SCRPTR, #6, EDT\$SL_LNO_EMPTY, EDT\$SL_CUR_SCRLN	1009		
	59	00000000G	00	D1	00168	21\$:	CMPL	EDT\$A_CSR_SCRPTR, R9	1010		
			19	12	0016F	21\$:	BNEQ	EDT\$A_CSR_SCRPTR, #1, EDT\$G_SCR_REBUILD	1016		
00000000G	00	00000000G	00	D4	00171	21\$:	CLRL	EDT\$A_CSR_SCRPTR, #1, EDT\$SSC_LNDEL	1026		
			06	28	00177	21\$:	MOVC3	EDT\$A_FST_AVLN, R0	1027		
			01	D0	00183	21\$:	MOVL	EDT\$A_FST_AVLN, R0	1028		
			59	DD	0018A	21\$:	PUSHL	EDT\$G_TRUN, 28\$	1035		
			01	FB	0018C	21\$:	CALLS	EDT\$G_SHF, R0	1042		
			00	D0	00193	21\$:	MOVL	EDT\$G_TI_WID, WIDTH	1046		
			A0	95	0019A	21\$:	TSTB	EDT\$G_TI_WID, WIDTH	1051		
			AC	13	0019D	21\$:	BEQL	EDT\$G_TI_WID, WIDTH	1057		
	58	00000000G	00	D1	0019F	21\$:	CMPL	EDT\$G_TI_WID, WIDTH	1059		
			B9	12	001A6	21\$:	BNEQ	EDT\$G_TI_WID, WIDTH	1062		
			A1	11	001A8	21\$:	BRB	EDT\$G_TI_WID, WIDTH	1064		
	6B		58	D0	001AA	22\$:	MOVL	NEW_SCRPTR, (R11)	1065		
	0D	04	AE	90	001AD	23\$:	MOVB	EDIT_CODE, 13(NEW_SCRPTR)	1066		
	08	18	AE	90	001B2	23\$:	MOVB	INDX, 8(NEW_SCRPTR)	1066		
	08	FF00	8F	B0	001B7	23\$:	MOVW	#65280, 11(NEW_SCRPTR)	1066		
	09		57	90	001BD	23\$:	MOVB	LEFT, 9(NEW_SCRPTR)	1066		
	77	00000000G	00	E8	001C1	23\$:	BLBS	EDT\$G_TRUN, 28\$	1066		
	50	00000000G	00	D0	001C8	23\$:	MOVL	EDT\$G_SHF, R0	1066		
52	00000000G	00	50	C1	001CF	23\$:	ADDL3	RO, EDT\$G_TI_WID, WIDTH	1066		
			18	AE	D5	001D7	TSTL	INDX	1066		
			05	12	001DA	23\$:	BNEQ	24\$	1066		
			10	AE	D4	001DC	CLRL	COL	1066		
			05	11	001DF	23\$:	BRB	25\$	1066		
	10	AE	02	A0	9E	001E1	24\$:	MOVAB	2(R0), COL	1066	
OC	AE	10	OC	AC	57	C3	001E6	25\$:	SUBL3	LEFT, REC_LEN, LEN	1066
		08	AE	08	BC47	9E	001EC	25\$:	MOVAB	@REC_ADDR[LEFT], TXT	1066
			56	FF	A7	9E	001F2	25\$:	MOVAB	-1(R7), RIGHT	1066
			6E	08	BE	9A	001F6	26\$:	MOVZBL	@TXT, CHAR	1066
				08	AE	D6	001FA	26\$:	INCL	TXT	1066
				OC	AE	D5	001FD	26\$:	TSTL	LEN	1066
					2E	15	00200	26\$:	BLEQ	27\$	1066
				10	AE	DD	00202	26\$:	PUSHL	COL	1066
				04	AE	DD	00205	26\$:	PUSHL	CHAR	1066
					02	FB	00208	26\$:	CALLS	#2, EDT\$FMT_CHWID	1066
00000000G	00		50	AE	C0	0020F	26\$:	ADDL2	COL, R0	1066	
	52		50	D1	00213	26\$:	CMPL	RO, WIDTH	1066	1064	
			18	14	00216	26\$:	PGTR	27\$	1066	1065	
			OC	AE	D7	00218	26\$:	DECL	LEN	1066	1066
				56	D6	0021B	26\$:	INCL	RIGHT	1066	1066
			10	AE	DD	0021D	26\$:	PUSHL	COL	1066	1066

		04	AE	DD	00220	PUSHL	CHAR	:	
00000000G	00		02	FB	00223	CALLS	#2, EDT\$FMT_CHWID	:	
10	AE		50	C0	0022A	ADDL2	R0, COL	:	
			C6	11	0022E	BRB	26\$	:	1067
		0C	AE	D5	00230	TSTL	LEN	:	1070
			0A	15	00233	BLEQ	28\$	:	
14	AE		01	D0	00235	MOVL	#1, NEED_ANOTHER_LINE	:	1076
57		01	A6	9E	00239	MOVAB	1(R6), LEFT	:	1077
			07	11	0023D	BRB	29\$	:	1070
		14	AE	D4	0023F	CLRL	NEED_ANOTHER_LINE	:	1094
	56	FF	8F	9A	00242	MOVZBL	#255, RIGHT	:	1095
0A	A8		56	90	00246	MOVB	RIGHT, 10(NEW_SCRPTR)	:	1098
		00000000G	00	D5	0024A	TSTL	EDT\$A_CUR_SCRPTR	:	1103
			07	12	00250	BNEQ	30\$	:	
00000000G	00		58	D0	00252	MOVL	NEW_SCRPTR, EDT\$A_CUR_SCRPTR	:	
		18	AE	D6	00259	INCL	INDX	:	1105
			FDB0	31	0025C	BRW	1\$	:	0825
	50	00000000G	00	9E	0025F	MOVAB	EDT\$Z_EOB_LN, R0	:	1108
	50	00000000G	00	D1	00266	CMPL	EDT\$A_WK_LN, R0	:	
			0B	12	0026D	BNEQ	32\$	:	
			5B	D5	0026F	TSTL	R11	:	
			07	12	00271	BNEQ	32\$	:	
00000000G	00		58	D0	00273	MOVL	NEW_SCRPTR, EDT\$A_EOB_SCRPTR	:	
	50	18	AE	D0	0027A	MOVL	INDX, R0	:	1149
				04	0027E	RET		:	
		50	D4	0027F	33\$:	CLRL	R0	:	1150
			04	00281	RET			:	

; Routine Size: 642 bytes, Routine Base: _EDT\$CODE + 0000

```

: 581      1151 1
: 582      1152 1 !<BLF/PAGE>

```


EDT\$SCRNEWINS
V04-000

EDT\$SCRNEWINS - insert a line on the screen
EDT\$\$\$SC_LNINS - insert a line on the screen

K 6
16-Sep-1984 01:38:16
14-Sep-1984 12:24:34

VAX-11 Bliss-32 V4.0-742
[EDT.SRC]SCRNEWINS.BLI;1

Page 16
(4)

: 584
: 585
: 586
1153 1 END
1154 1
1155 0 ELUDOM

! of module EDT\$SCRNEWINS

PSECT SUMMARY

Name	Bytes	Attributes
_EDT\$CODE	642	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$255\$DUA28:[EDT.SRC]EDT.L32;1	377	54	14	40	00:00.2
-\$255\$DUA28:[EDT.SRC]PSECTS.L32;1	2	1	50	7	00:00.1

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACEBACK/LIS=LIS\$:SCRNEWINS/OBJ=OBJ\$:SCRNEWINS MSRC\$:SCRNEWINS.BLI/UPDATE=(ENH\$:S
CRNEWINS)

: Size: 642 code + 0 data bytes
: Run Time: 00:34.1
: Elapsed Time: 00:39.4
: Lines/CPU Min: 2030
: Lexemes/CPU-Min: 8956
: Memory Used: 229 pages
: Compilation Complete

0139 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

